Brad Abrams

Designing Microsoft® .NET Class Libraries

June 2004

Course Objectives

- Provide guidance for you to design the next generation of platform APIs
- After successfully completing this course, you will:
  - Understand the breadth of issues .NET library designers face
  - Be equipped to build great .NET libraries
  - Have access to resources to help you deal with ongoing issues

Introductions

About the Instructor

Student Introductions (via e-mail list—*designdn*)

- Name
- Title/function
- Job responsibility
- Product group and experience
- Expectations for the course

Course Materials

- Slides

http://ddwww/SpecTool/Documents/Milestone-independent/DesignGuidelines/MSTE-Designing.NETClassLibraries.ppt

- Design Guidelines Document

http://ddwww/SpecTool/Documents/Milestone-independent/DesignGuidelines/designguidelines.doc

Facilities

- Class Hours
  - 9:00 A.M. start
  - 11:45–12:45 lunch (not provided)
  - 5:15–5:45 P.M. finish
- Meals
  - Buildings 17 and 34 have cafeterias nearby
- Restrooms
- Phones

Audience and Prerequisites

- Audience
  - Developers
  - Program managers
  - Others designing .NET class libraries
- Before you take this course you should have the following knowledge and skills:
  - Working knowledge of the Microsoft .NET Framework and a .NET language (about 0.5–1 year)
  - Objected-oriented (OO) design experience

What We Will Cover—Thursday

Time		Title	Instructor
9:00 A.M.	0	Setting the Stage	BradA
9:15	1	Naming Conventions	BradA
10:15		Break	
10:30	2	Rich Type System	BradA
11:45		Lunch	
12:00 P.M.		(bonus)Usability Presentation	StevenCl
12:45		After Lunch Wake Up: Usability Videos	
1:00 P.M.	3	Designing Progressive APIs	KOwalina
1:30	4	Member Types	BradA
2:30	5	Designing Inheritance Hierarchies	BradA
3:30		Break	
3:45	6	CLR Performance Tips	RicoM
4:45	7	Designing for Managed-Memory World	Brada

What We Will Cover—Friday

Time		Title	Instructor
9:00 A.M.	8	Understanding Interop	SonjaKe
10:15		Break	
10:30	9	Versioning	RLander
11:00	10	Packaging, Assemblies, and Namespaces	MiMurray
11:45		Lunch	
12:00 P.M.		Bonus: FxCop in Depth	Jeffva, MiMurray
12:45 P.M.		After Lunch Wake Up: FxCop Demo	
1:00 P.M.	11	Enabling Development Tools	BrianPe
2:15	12	Security	SLange
3:45		Break	
4:00	12	.NET Q&A	v-JeffrR and Team

Lesson 0: Setting the Stage

- After successfully completing this lesson, you will:
  - Be exposed to basic terminology
  - Understand the why behind Microsoft WinFX™
  - Appreciate your part in the WinFX goals
  - Recognize the role of guidelines
  - Be aware of the first principles
  - Know your role in the *multi-language* runtime

Terminology

- Managed Code: Code where execution is closely managed by the common language runtime (CLR) —for example, code produced by C# and Microsoft Visual Basic® .NET compilers
- Unmanaged Code: All other code in the world
- .NET Libraries: Reusable managed code classes that are intended to be used by third-party developers (not the output of TLBImp)
- Applications: Software with which end users interact

The Microsoft Business Model

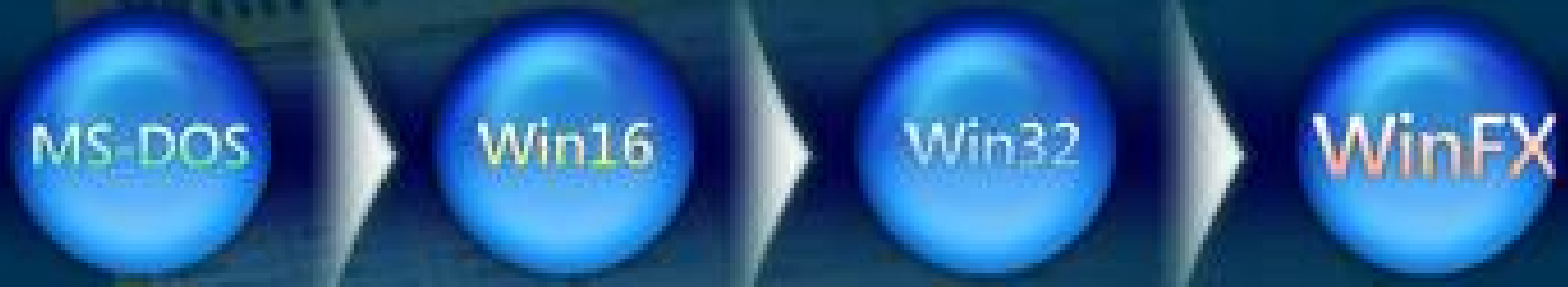
Platform Advances

The Microsoft Business Model

The Microsoft Business Model

The Microsoft Business Model

Microsoft Platform History



WinFX

- Builds on the .NET Framework
- Is a well-structured programming framework for Microsoft Windows®
- Continues commitment to backward compatibility

WinFX "Prime Directive"

First-class managed interfaces to all broadly used system technologies

About Design Guidelines

- Often Subjective
  - But benefits of consistency aren't subjective
- Contentious by Nature
- Intrinsic Value in Conformance
  - Developers' knowledge transfers
  - High-level design tools can be built
  - Time-tested patterns
- About Conveying Ideas to the Developers

About .NET Design Guidelines

- Easier to Use
  - Raised abstraction level
  - Rapid Application Development (RAD) / component-oriented
- Single API Targets Multiple Users
  - Range of languages: Visual Basic, C#, C++, third-party
  - Range of styles: RAD, systems, etc.
  - Consistent availability of functionality
    - Single platform
    - More headroom
  - Increased complexity for increased control
    - Example: Async IO

First Principles

- What does your library look like to program against?
- Design in reverse:
  - Write the three lines of code the developer will have to write, then model the APIs around making that possible
  - Example: `Console.WriteLine()`
- Understand what it is like from other languages as well

Common Language Specification (CLS)

- Ensures components can be accessed from any programming language
- Is fundamentally a compromise between languages and component providers
- Adhering to the CLS broadens your user base
- C# does automatic checking
  - Visual Basic and MC++ (v1) do not check

```
[assembly: CLSCompliant(true)]
```

What We Will NOT Cover

- Introduction to .NET or C#
- Source Code Conventions
- Data Structures and Algorithms
- Introduction to Object-Oriented Design

Lesson 0 Summary

- WinFX starts a virtuous cycle
- The WinFX "Prime Directive" is: First-class managed interfaces to all broadly used system technologies
- Guidelines are often contentious but valuable
- The CLS is important and easy to follow